GREEDY MATCHING IN BIPARTITE RANDOM GRAPHS

NICK ARNOSTI, COLUMBIA BUSINESS SCHOOL

THIS VERSION: APRIL 2021

Abstract. This paper studies the performance of greedy matching algorithms on bipartite graphs $G = (\mathcal{J}, \mathcal{D}, \mathcal{E})$. We focus primarily on three classical algorithms: RANDOM-EDGE, which sequentially selects random edges from $\mathcal{E}$; RANDOM-VERTEX, which sequentially matches random vertices in $\mathcal{J}$ to random neighbors, and RANKING, which generates a random priority order over vertices in $\mathcal{D}$ and then sequentially matches random vertices in $\mathcal{J}$ to their highest-priority remaining neighbor. Prior work has focused on identifying the worst-case approximation ratio for each algorithm. This guarantee is highest for RANKING, and lowest for RANDOM-EDGE. Our work instead studies the *average* performance of these algorithms when the edge set $\mathcal{E}$ is random.

Our first result compares RANDOM-VERTEX and RANDOM-EDGE, and shows that on average, RANDOM-VERTEX produces more matches. This result holds for finite graphs (in contrast to previous asymptotic results), and also applies to "many to one" matching in which each vertex in $\mathcal{D}$ can match with multiple vertices in $\mathcal{J}$.

Our second result compares RANDOM-VERTEX and RANKING, and shows that the better worst-case guarantee of RANKING does not translate into better average performance. In "one to one" settings where each vertex in $\mathcal{D}$ can match with only one vertex in $\mathcal{J}$, the algorithms result in the same number of matches. When each vertex in $\mathcal{D}$ can match with two vertices in $\mathcal{J}$, RANDOM-VERTEX produces *more* matches than RANKING.

1. Introduction

This paper studies the performance of greedy algorithms for many-to-one bipartite matching. While bipartite matching has many applications, we adopt the terminology of scheduling "jobs" on different "days." Although maximum matchings can be found in polynomial time, there has been considerable interest in understanding the performance of simple greedy algorithms. Performance is traditionally measured by the worst case ratio between the size of the matching produced by the algorithm and the size of a maximum matching. No deterministic greedy algorithm can provide a guarantee above 1/2 (Karp et al. 1990), so attention has focused on randomized greedy algorithms.

One natural algorithm considers edges in a random order. We call this RANDOM-EDGE; it is referred to as "simple case algorithm" by Tinhofer (1984), and "greedy" by Dyer and Frieze (1991). Another algorithm schedules jobs sequentially, selecting among feasible days uniformly at random. We call this RANDOM-VERTEX; it is referred to as "modified random greedy" by Aronson et al. (1995) and "random" by Karp et al. (1990), and most subsequent work adopts one of these names. A third alternative sequentially schedules jobs using a universal ordering over days. This approach was introduced by Karp et al. (1990) under the name RANKING, which we also adopt.[1]

Dyer and Frieze (1991) show that the worst-case guarantee of RANDOM-EDGE is only 1/2. A series of papers (discussed in more detail below) have established increasingly tight bounds for the guarantees offered by RANDOM-VERTEX and RANKING; the tightest bounds of which we are

[1]Although we adopt the name RANKING, there is an important difference between our algorithm and that studied by Karp et al. (1990): whereas they assume that the order in which jobs are considered is adversarial, most subsequent work (including our own) assumes that this order is random. Similarly, the "random" algorithm of Karp et al. (1990) assumes an adversarial order of jobs, whereas "modified random greedy" randomizes this order. As we discuss in more detail below, randomizing arrival order allows for strictly better guarantees than those proved by Karp et al. (1990).

	Worst Case Ratio
RANDOM-EDGE	0.5
RANDOM-VERTEX	[0.639, 0.646]
RANKING	[0.696, 0.724]

Table 1. Worst-case approximation ratios for one-to-one bipartite matching, from prior work. Although RANKING offers the best guarantee, on some graphs it is outperformed by RANDOM-VERTEX and RANDOM-EDGE (see Figure 1). We study the average performance of these algorithms. In one-to-one settings, RANKING and RANDOM-VERTEX have identical performance, and both outperform RANDOM-EDGE. RANDOM-VERTEX continues to outperform RANDOM-EDGE in many-to-one settings (Theorem 1), and outperforms RANKING in two-to-one settings (Theorem 2).

aware are presented in Table 1. These bounds establish that of these algorithms, RANKING offers the best guarantee, and RANDOM-EDGE the worst. Largely based on these worst-case bounds, graduate students who study online matching learn that RANKING is a better algorithm for online matching than RANDOM-VERTEX. Meanwhile, the following heuristic argument "explains" the poor guarantee of RANDOM-EDGE: it disproportionately schedules high-degree jobs up front, even though low-degree jobs intuitively face the greatest risk of going unassigned.

This reasoning and the bounds in Table 1 notwithstanding, there are in fact graphs where RANDOM-VERTEX outperforms RANKING, and both are outperformed by RANDOM-EDGE (see Figure 1). How common are these cases, and what is true on a "typical" graph? Does worst-case analysis paint a representative picture of these algorithms' performance?

We tackle these questions by comparing these algorithms' *average* performance across all graphs where jobs have a specified degree sequence. Our first result is that the aforementioned intuition for the poor performance of RANDOM-EDGE can be made rigorous: for any specified degree sequence, the average performance of RANDOM-VERTEX exceeds that of RANDOM-EDGE. Our second result establishes that the apparent gap between RANDOM-VERTEX and RANKING does not transfer to average performance: the two algorithms are equivalent for one-to-one matching, and RANDOM-VERTEX produces *more* matches when two jobs can be assigned to each day.

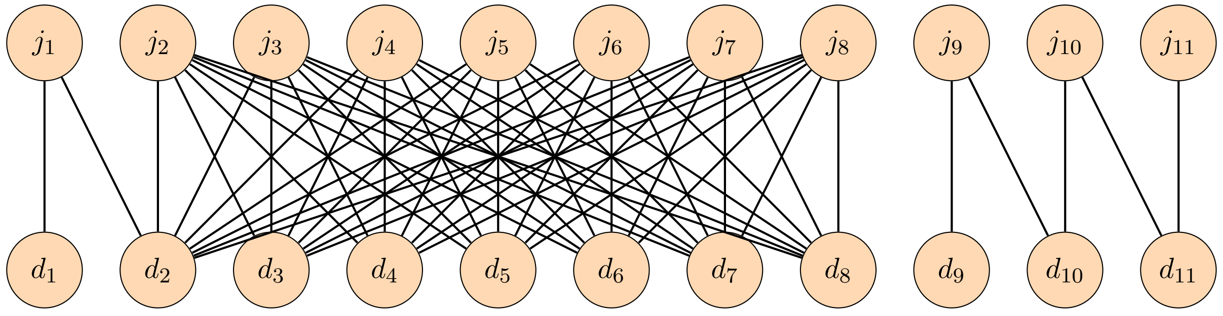
In Section 1.1 we introduce our notation and formally define the algorithms RANDOM-EDGE, RANDOM-VERTEX, and RANKING. Section 1.2 describes what is known about the worst-case performance of these algorithms. Section 1.3 states our results, and compares these results to existing average-case analyses. We prove our main results in Sections 2 and 3.

1.1. **Preliminaries: Notation and Definitions.**

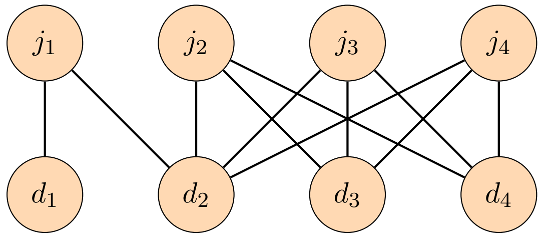
We adopt the terminology of assigning a set of jobs $\mathcal{J}$ to a set of days $\mathcal{D}$. Each job has associated *scheduling* constraints, which are captured by the edge set $\mathcal{E} \subseteq \mathcal{J} \times \mathcal{D}$: the edge $(j, d) \in \mathcal{E}$ if and only if job j can be scheduled for day d. Each day d has an associated *capacity* constraint $C_d \in \mathbb{N}$, indicating the maximum number of jobs that can be scheduled on that day. We adopt the following definitions.

Definition 1. Fix $G = (\mathcal{J}, \mathcal{D}, \mathcal{E})$, and $\mathbf{C} \in \mathbb{N}^{\mathcal{D}}$.

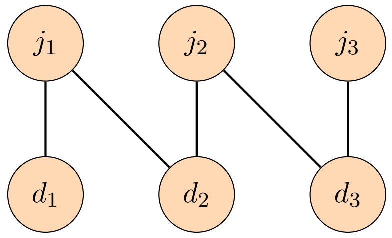
- A **matching** is a set of edges $\mathcal{M} \subseteq \mathcal{J} \times \mathcal{D}$.
- A matching $\mathcal{M}$ is **feasible** (with respect to $\mathcal{E}$ and $\mathbf{C}$) if $\mathcal{M} \subseteq \mathcal{E}$, and in the graph $(\mathcal{J}, \mathcal{D}, \mathcal{M})$, each $j \in \mathcal{J}$ has at most one neighbor and each $d \in \mathcal{D}$ has at most C_d neighbors.
- A matching $\mathcal{M}$ is **maximal** (with respect to $\mathcal{E}$ and $\mathbf{C}$) if it is feasible, and for all $e \in \mathcal{E} \backslash \mathcal{M}$, $\mathcal{M} \cup \{e\}$ is not feasible.
- A matching $\mathcal{M}$ is **maximum** (with respect to $\mathcal{E}$ and $\mathbf{C}$) if it is feasible, and no feasible matching has more edges.


j_1 j_2 j_3 j_4 j_5 j_6 j_7 j_8 j_9 j_{10} j_{11}
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11}

(A) Expected unassigned jobs is < 0.704 under RANDOM-EDGE, $17/24 \approx 0.708$ under RANDOM-VERTEX, and $17/24 + 1/48 \approx 0.729$ under RANKING.



(B) Expected unassigned jobs:
$5/24 \approx 0.21$ under RANKING,
$8/33 \approx 0.24$ under RANDOM-EDGE, and
$1/4 = 0.25$ under RANDOM-VERTEX.



(C) Expected unassigned jobs:
$11/24 \approx 0.46$ under RANDOM-VERTEX,
$8/15 \approx 0.53$ under RANDOM-EDGE, and
$13/24 \approx 0.54$ under RANKING.

FIGURE 1. Comparisons between RANKING, RANDOM-VERTEX, RANDOM-EDGE. Although worst-case analysis suggests that RANKING performs best and RANDOM-EDGE worst (see Table 1), this ordering is reversed on the graph in 1a. For any pair of these algorithms, one performs better on the graph in 1b, and the other performs better on the graph in 1c.

This paper considers greedy algorithms which are parameterized by a complete order $\succ^E$ on $\mathcal{J} \times \mathcal{D}$. These algorithms start with $\mathcal{M} = \emptyset$, and in the order given by $\succ^E$, sequentially add edges to $\mathcal{M}$ so long as the result remains feasible. We let $\mathcal{M}(\mathcal{E}, \mathbf{C}, \succ^E)$ denote the resulting matching.

Note that the order $\succ^E$ is *non-adaptive*, in that earlier matches do not affect the order in which later edges are considered. This family of algorithms has been referred to as the ”query commit problem” by Goel and Tripathi (2012) and the “oblivious matching model” by Tang et al. (2020).[2]

Although there are many ways to generate the order $\succ^E$, several special cases are of particular interest. The RANDOM-EDGE algorithm selects the order $\succ^E$ uniformly at random. A different set of greedy algorithms consider jobs sequentially. They arise naturally in the context of online matching, where the set of jobs is gradually revealed, and each job must be (irrevocably) scheduled upon arrival. These algorithms are parameterized by an order $\succ^{\mathcal{J}}$ in which jobs will be considered, and a collection of rankings $\succ^{\mathcal{D}} = \{\succ^{\mathcal{D}}_j\}_{j \in \mathcal{J}}$, where $\succ^{\mathcal{D}}_j$ is a ranking over days used when scheduling job j. Formally, we define $\mathrm{VI}(\succ^{\mathcal{J}}, \succ^{\mathcal{D}})$ to be the order $\succ^E$ satisfying

$$(j, d) \succ^E (j', d') \Leftrightarrow j \succ^{\mathcal{J}} j' \text{ or } j = j' \text{ and } d \succ^{\mathcal{D}}_j d'.$$

Goel and Tripathi (2012) and Tang et al. (2020) refer to these as “vertex-iterative algorithms”, and we choose the letters VI to align with this terminology. Both RANDOM-VERTEX, and RANKING are special cases of vertex-iterative algorithms. RANDOM-VERTEX selects $\succ^{\mathcal{J}}$ and the orders $\succ^{\mathcal{D}}_j$

[2]It is also closely related to the class of “randomized priority algorithms” described by Borodin et al. (2003) for a class of scheduling problems.

independently and uniformly at random. RANKING selects the order $\succ^{\mathcal{J}}$ and an order $\succ$ on $\mathcal{D}$ independently and uniformly at random, and sets $\succ_j^{\mathcal{D}}=\succ$ for all $j \in \mathcal{J}$.

1.2. **Prior Work: Worst Case Analysis.** Most existing analysis assumes one-to-one matching ($C_d = 1$ for all d), and studies an algorithm's worst-case approximation ratio: that is, the largest number α such that the expected size of the matching produced by the algorithm is always at least α times the size of a maximum matching.

Dyer and Frieze (1991) show that for RANDOM-EDGE, no guarantee greater than 1/2 is possible. Karp et al. (1990) study vertex-iterative algorithms in a setting where both the scheduling constraints $\mathcal{E}$ and the job arrival order $\succ^{\mathcal{J}}$ are adversarial, and only the orders $\succ_j^{\mathcal{D}}$ can be chosen. They show that if the $\succ_j^{\mathcal{D}}$ are iid and uniformly distributed (the "random" algorithm) then no guarantee greater than 1/2 is possible. They propose instead choosing the $\succ_j^{\mathcal{D}}$ to be *identical* (the "ranking" algorithm), and show that the expected size of the resulting matching is at least $1 - 1/e \approx 0.63$ times the size of the maximum matching.[3]

Subsequent work has assumed that the arrival order $\succ^{\mathcal{J}}$ is drawn uniformly at random, which corresponds to the algorithms RANDOM-VERTEX and RANKING as described in this paper. This enables stronger guarantees. Aronson et al. (1995) prove that RANDOM-VERTEX (which they name "modified random greedy") offers a guarantee of $1/2 + 2.5 \times 10^{-6}$. This was state-of-the-art until Poloczec and Szegedy (2012) proved[4] $1/2 + 1/256$. In recent work, Tang et al. (2020) prove a guarantee of 0.639, and establish an upper-bound of 0.646. For RANKING, Mahdian and Yan (2011) establish a guarantee of 0.696 and Karande et al. (2011) provide an upper bound of 0.727 which has since been improved to 0.724 by Chan et al. (2018).[5] Table 1 summarizes our knowledge of the worst-case performance guarantee for each algorithm.

There is a natural intuition that it is best to start by scheduling low-degree jobs. A variant of RANDOM-VERTEX which does this was proposed by Tinhofer (1984) and subsequently dubbed "MINGREEDY." Frieze et al. (1995) show that this algorithm performs well on random cubic graphs, but somewhat surprisingly, Besser and Poloczek (2017) show that its worst-case guarantee is the trivial one of 1/2.

1.3. **Our Results: Average Case Analysis.** This paper considers the performance of these greedy algorithms when the scheduling constraints $\mathcal{E}$ are random, as defined below.

Definition 2. For any $\mathcal{J}, \mathcal{D}$ and $\bar{\mathbf{N}} \in \{0, 1, \ldots, |\mathcal{D}|\}^{\mathcal{J}}$, define $\psi(\bar{\mathbf{N}})$ to be the uniform distribution over edge sets $\mathcal{E}$ for which each $j \in \mathcal{J}$ has exactly $\bar{\mathbf{N}}_j$ neighbors.

Note that one can sample from $\psi(\bar{\mathbf{N}})$ by allowing each $j \in \mathcal{J}$ to independently select $\bar{N}_j$ neighbors in $\mathcal{D}$ uniformly at random. This family of bipartite random graphs has previously been studied in the context of cuckoo hashing (Dietzfelbinger et al. 2010, Fountoulakis and Panagiotou 2012, Frieze et al. 2018).

Our first result establishes that for this family of random graphs, the size of the matching generated by RANDOM-VERTEX stochastically dominates the size of the matching generated by RANDOM-EDGE.

Theorem 1. *Fix $\mathcal{J}, \mathcal{D}$, $\mathbf{C} \in \mathbb{N}^{\mathcal{D}}$ and $\bar{\mathbf{N}} \in \{0, 1, \ldots |\mathcal{D}|\}^{\mathcal{J}}$. If $\mathcal{E}$ is drawn from $\psi(\bar{\mathbf{N}})$, then $\forall k \in \mathbb{N}$,*

$$\mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANDOM-VERTEX})| \geq k) \geq \mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANDOM-EDGE})| \geq k).$$

[3]Although the results claimed by Karp et al. (1990) are correct, Goel and Mehta (2008) observed and corrected an error in their original analysis.

[4]Chan et al. (2018) comment that there are several gaps in their proof.

[5]It is possible to generalize RANDOM-VERTEX and RANKING to general (non-bipartite) graphs. Of course, this results in lower worst-case performance. Goel and Tripathi (2012) claimed a guarantee of 0.56 for RANKING, but retracted the paper after errors were found. Chan et al. (2018) establish a guarantee of 0.523 for RANKING, and recent work by Tang et al. (2020) establishes a guarantee of 0.531 for both RANDOM-VERTEX and RANKING.

Note that this immediately implies the corresponding result when the values $\bar{\mathbf{N}}_j$ are drawn from an arbitrary joint distribution (i.e. the case of bipartite Erdös-Renyi graphs). Appendices A.1 and A.2 establish that this result continues to hold in extensions where jobs have different "sizes" and the neighbors of each job are drawn from a non-uniform distribution.

Our second result says while RANKING offers better worst-case guarantees than RANDOM-VERTEX, this does not result in better average performance. In fact, when each day has capacity $C_d = 1$ (as much of the literature assumes), RANKING and RANDOM-VERTEX have identical performance, and when each day has capacity $C_d = 2$, RANDOM-VERTEX produces a stochastically larger matching.

Theorem 2. *Fix $\mathcal{J}, \mathcal{D}$, and $\bar{\mathbf{N}} \in \{0, 1, \ldots |\mathcal{D}|\}^{\mathcal{J}}$, and let $\mathcal{E}$ be drawn from $\psi(\bar{\mathbf{N}})$.*
If $C_d = 1$ for all $d \in \mathcal{D}$, then for all $k \in \mathbb{N}$,

$$\mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANDOM-VERTEX})| \geq k) = \mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANKING})| \geq k).$$

If $C_d = 2$ for all $d \in \mathcal{D}$, then for all $k \in \mathbb{N}$,

$$\mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANDOM-VERTEX})| \geq k) \geq \mathbb{P}(|\mathcal{M}(\mathcal{E}, \mathbf{C}, \text{RANKING})| \geq k).$$

In fact, when $C_d = 1$ for all days, all vertex-iterative algorithms that consider jobs in the same order have equivalent performance: the preferences of individual jobs do not affect the distribution of the size of the resulting matching. Therefore, Theorem 1 implies that any vertex-iterative algorithm that considers jobs in a random order results in a stochastically larger matching than RANDOM-EDGE.

The intuition for the second claim is that when jobs use the same order, highly-ranked days tend to be chosen by many jobs. As a result, they reach their capacity quickly. This increases the probability that later jobs go unassigned. We conjecture that RANDOM-VERTEX continues to yield a stochastically larger matching when all days have identical capacity $C_d = C > 2$.[6] At the end of Section 3, we present simulations supporting this conjecture, and explain why our current proof does not extend to this case.

1.4. **Prior Work: Average Case Analysis.**

We are not the first paper to conduct average-case analysis of these greedy matching algorithms. Dyer et al. (1993) study the asymptotic performance of RANDOM-EDGE and RANDOM-VERTEX (which they refer to as "GREEDY" and "MODIFIED GREEDY," respectively) on two families: sparse Erdös-Renyi random graphs, and random labeled trees. For both families, they show that as the number of vertices grows large, the expected matching size is larger under RANDOM-VERTEX than under RANDOM-EDGE. Although our results are technically incomparable because we study bipartite random graphs, Theorem 1 is "stronger" than this result in several ways.

- It holds for graphs of arbitrary size, rather than only asymptotically.
- It allows vertices in $\mathcal{D}$ to have arbitrary capacities, rather than assuming $C_d = 1\ \forall d$.
- It allows for an arbitrary degree distribution among vertices in $\mathcal{J}$, rather than the binomial distribution that arises in Erdös-Renyi random graphs.

To our knowledge, the only other paper that attempts a non-asymptotic average-case comparison of RANDOM-EDGE and RANDOM-VERTEX is Tinhofer (1984). That analysis is restricted to the case of Erdös-Renyi random graphs, and a comparison between the algorithms is provided only when each edge is present in $\mathcal{E}$ with probability exceeding 1/2.

[6]When days have *unequal* capacities, it is possible for RANKING to outperform RANDOM-VERTEX. Consider an example with two days. For half of jobs, both days are feasible; for the other half, only one (randomly selected) day is feasible. If $C_{d_1} = \frac{1}{4}|\mathcal{J}|$ and $C_{d_2} = \frac{3}{4}|\mathcal{J}|$, then as $|\mathcal{J}| \to \infty$, the expected fraction of unassigned jobs approaches 1/12 under RANKING, 1/8 under RANDOM-VERTEX, and 0 under a maximum matching.

Lemma 4 in Mastin and Jaillet (2018) notes that on Erdös-Renyi random graphs, the performances of RANKING and RANDOM-VERTEX are identical. The first part of Theorem 2 generalizes their result by allowing for an arbitrary degree distribution among vertices in $\mathcal{J}$. Furthermore, our proof clarifies that this equivalence relies only on randomness of the edge set $\mathcal{E}$, and not on randomness of the algorithms themselves. We are unaware of any results similar to the second part of Theorem 2, which compares RANKING and RANDOM-VERTEX when multiple jobs can be assigned to each day.

2. Proof of Theorem 1

The key intuition underlying the proof of Theorem 1 is that it is best to start by scheduling jobs with few feasible days, because jobs with many feasible days can likely be scheduled later. RANDOM-EDGE does the *opposite*: at each step, jobs with more feasible days are more likely to be scheduled. Meanwhile, RANDOM-VERTEX schedules jobs in a random order, and thus is more likely than RANDOM-EDGE to schedule low-degree jobs.

To formalize this idea, Section 2.1 describes Markov chains corresponding to procedures RANDOM-EDGE and RANDOM-VERTEX. In each Markov chain, the matching $\mathcal{M}$ is constructed iteratively, starting with $\mathcal{M}_0 = \emptyset$ and adding one edge at a time. Rather than revealing the random edges $\mathcal{E}$ initially, these chains start with $\mathcal{E}_0 = \emptyset$ and reveal an edge $e = (j, d) \in \mathcal{E}$ only when either i) j is matched, or ii) d reaches its capacity. Potential edges between unassigned jobs and days with excess capacity remain unobserved. The size of the matching generated by each procedure corresponds to the number of steps before the full edge set $\mathcal{E}$ is revealed (implying that no more matches can be formed).

Section 2.2 couples these chains such that the number of unrevealed edges under RANDOM-VERTEX is always weakly larger than the corresponding number under RANDOM-EDGE. Whenever the number of jobs with at least k unrevealed edges is identical in the two chains, the coupling ensures that if the next job assigned under RANDOM-VERTEX has at least k unrevealed edges, then so does the next job assigned under RANDOM-EDGE. Figure 2 provides an illustration of each chain, and the coupling between them.

2.1. Proof Step 1: Markov Chain Description.

Throughout, we fix a number of feasible days $\bar{N}_j$ for each job j and capacities C_d for each day d. For both RANDOM-VERTEX and RANDOM-EDGE, we start with an empty edge set and matching $\mathcal{E}_0 = \mathcal{M}_0 = \emptyset$. At each step t, we add one edge between an unscheduled job and a day with idle capacity to the matching $\mathcal{M}_t$ (if no such edge exists, then our procedure has finished). We also reveal any edges involving matched jobs or days with no remaining capacity and add these to the edge set $\mathcal{E}_t$. This ensures that unrevealed edges are precisely those that can be feasibly added to $\mathcal{M}_t$. In other words, for any $e = (j, d)$,

$$\mathcal{M}_t \cup \{e\} \text{ is feasible} \Leftrightarrow e \in \mathcal{E} \backslash \mathcal{E}_t.$$

Given a set of edges $\mathcal{E}_t$ and a feasible matching $\mathcal{M}_t \subseteq \mathcal{E}_t$, define

(1) $I_d(\mathcal{M}_t) = C_d - |\{e \in \mathcal{M}_t : e = (j, d) \text{ for some } j\}|$ Idle capacity on day d.

(2) $N_j(\mathcal{E}_t) = \bar{N}_j - |\{e \in \mathcal{E}_t : e = (j, d) \text{ for some } d\}|$ Number of unknown feasible days for job j.

(3) $F_k(\mathcal{E}_t) = |\{j : N_j(\mathcal{E}_t) = k\}|$ Number of jobs with k unknown feasible days.

We now describe procedures for adding edges to $\mathcal{M}_t$ corresponding to each algorithm. For each procedure, step $t + 1$ consists of four stages. For RANDOM-VERTEX these stages are as follows.

V1 **Reveal the next job to be assigned.** Select a uniform random job j_{t+1} from the set of unassigned jobs for which at least one feasible day has idle capacity:

$$\mathbb{P}(j_{t+1} = j | \mathcal{M}_t, \mathcal{E}_t) = \frac{\mathbf{1}(N_j(\mathcal{E}_t) > 0))}{\sum_{j'} \mathbf{1}(N_{j'}(\mathcal{E}_t) > 0)}. \tag{4}$$

V2 **Reveal feasible days for j_{t+1}.** Select $D_{t+1} \subseteq \{d : I_d(\mathcal{M}_t) > 0\}$ uniformly at random among subsets of size $N_{j_{t+1}}(\mathcal{M}_t)$, and define

$$\mathcal{A}_{t+1} = \{(j_{t+1}, d) : d \in D_{t+1}\}. \tag{5}$$

V3 **Assign j_{t+1} to a feasible day.** Select d_{t+1} uniformly at random from D_{t+1}, and set

$$\mathcal{M}_{t+1} = \mathcal{M}_t \cup (j_{t+1}, d_{t+1}). \tag{6}$$

Note that steps V2 and V3 imply that d_{t+1} is selected uniformly at random from days with idle capacity. That is,

$$\mathbb{P}(d_{t+1} = d | \mathcal{M}_t, \mathcal{E}_t, j_{t+1}) = \frac{\mathbf{1}(I_d(\mathcal{M}_t) > 0)}{|\{d : I_d(\mathcal{M}_t) > 0\}|}. \tag{7}$$

V4 **If d_{t+1} has no remaining capacity, reveal neighbors of d_{t+1} in $\mathcal{E}$.** Draw independent binary random variables $\{B_{(t+1)j}\}_{j \in \mathcal{J}}$, with

$$\mathbb{P}(B_{(t+1)j} = 1 | \mathcal{E}_t, \mathcal{M}_t, j_{t+1}, D_{t+1}, d_{t+1}) = \frac{N_j(\mathcal{E}_t)}{|\{d : I_d(\mathcal{M}_t) > 0\}|} \mathbf{1}(j \neq j_{t+1}) \mathbf{1}(I_{d_{t+1}}(\mathcal{M}_{t+1}) = 0). \tag{8}$$

Set

$$\mathcal{B}_{t+1} = \{(j, d_{t+1}) : B_{(t+1)j} = 1\} \tag{9}$$

$$\mathcal{E}_{t+1} = \mathcal{E}_t \cup \mathcal{A}_{t+1} \cup \mathcal{B}_{t+1}. \tag{10}$$

Meanwhile, for RANDOM-EDGE we have the following.

E1 **Reveal the job j_{t+1} which has the highest-ranking edge in $\mathcal{E} \backslash \mathcal{E}_t$** (according to $\succ^E$):

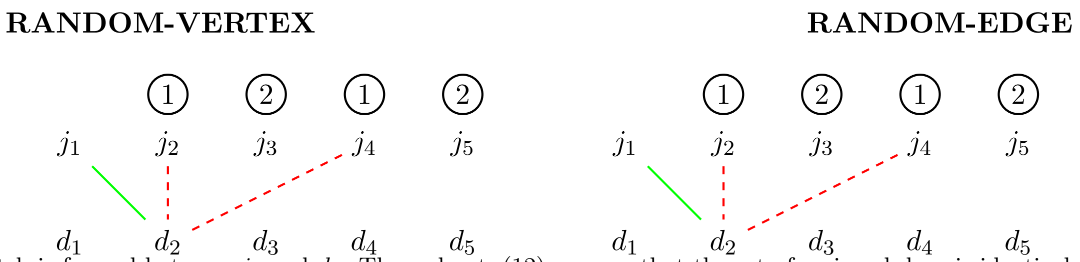
$$\mathbb{P}(j_{t+1} = j | \mathcal{M}_t, \mathcal{E}_t) = \frac{N_j(\mathcal{E}_t)}{\sum_{j'} N_{j'}(\mathcal{E}_t)}. \tag{11}$$

E2 **Reveal feasible days for j_{t+1}.** (Identical to V2).
E3 **Assign j_{t+1} to a feasible day.** (Identical to V3).
E4 **If d_{t+1} has no remaining capacity, reveal neighbors of d_{t+1} in $\mathcal{E}$.** (Identical to V4).

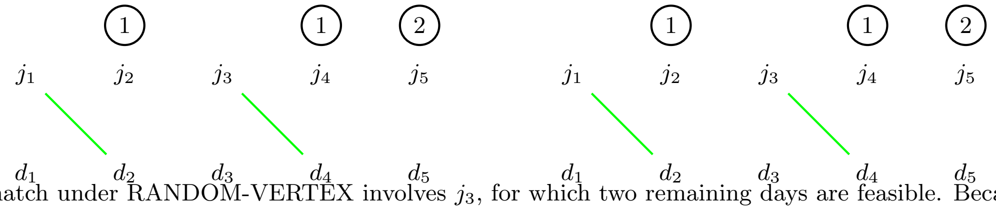
Note that the only difference between the procedures is that step V1 selects an unassigned job uniformly at random among those with at least one feasible day remaining, whereas step E1 selects among these jobs *in proportion to the number of feasible days remaining.*

Both procedures terminate when no more edges can be added – that is, $F_k(\mathcal{E}_t) = 0$ for all $k \geq 1$.

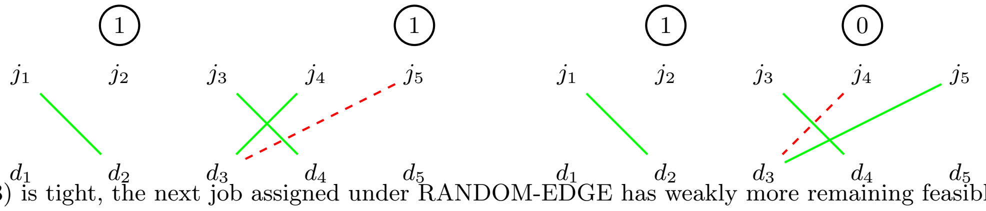
2.2. **Proof Step 2: Coupling.** In what follows, we use the superscript V to denote the chain corresponding to RANDOM-VERTEX, and E to denote the chain corresponding to RANDOM-EDGE. In other words, $\mathcal{M}_t^V, \mathcal{E}_t^V$ refer to the (random) edges and matching generated after t steps of V1-V4, and $\mathcal{M}_t^E, \mathcal{E}_t^E$ to the (random) edges and matching generated after t steps of E1-E4. For $X \in \{V, E\}$, let j_{t+1}^X be the job assigned at step $t+1$, let D_{t+1}^X be the days to which this job could feasibly be assigned, and let d_{t+1}^X be the day to which this job is actually assigned. Let τ^X be the time satisfying $F_k(\mathcal{E}_{\tau^X}^X) = 0$ for all $k \geq 1$. In other words, τ^X is the time at which it is no longer possible to add feasible edges to $\mathcal{M}_t^X$. We also recall (1), which defines $I(\mathcal{M}) \in \mathbb{N}^{\mathcal{D}}$ to be a vector indicating the idle capacity for each day, given matching $\mathcal{M}$. The following Lemma immediately implies Theorem 1.



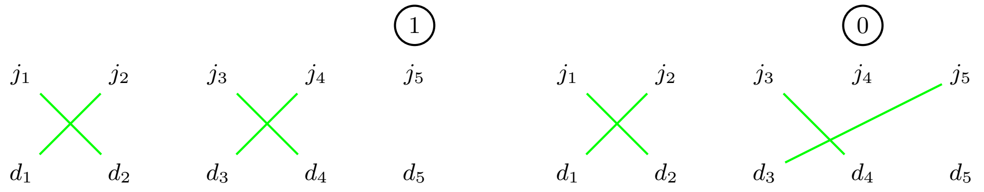
A match is formed between j_1 and d_2. Throughout, (12) ensures that the set of assigned days is identical for the two chains. Because (16) is tight, the coupling ensures that the revealed (red dashed) edges for the two chains coincide. It is revealed that d_2 was also feasible for j_2 and j_4.



The next match under RANDOM-VERTEX involves j_3, for which two remaining days are feasible. Because (13) is tight, the coupling ensures that the next match under RANDOM-EDGE also involves a job for which two remaining days are feasible. Because (16) is tight, the (empty) sets of revealed edges coincide. Jobs j_2 and j_4 are more likely to go unassigned than j_5, for which two of $\{d_1, d_3, d_5\}$ are feasible.



Because (13) is tight, the next job assigned under RANDOM-EDGE has weakly more remaining feasible days (see Lemma 2). The next match under RANDOM-VERTEX involves j_4, whereas under RANDOM-EDGE it involves j_5. For the first time, ϕ is not the identity: it permutes j_4 and j_5. Job j_4 can no longer be feasibly assigned under RANDOM-EDGE.



RANDOM-EDGE must match j_2. RANDOM-VERTEX randomly selects which of j_2 and j_5 to match. Job j_2 is assigned to d_1, and it is revealed that d_1 was not feasible for j_5. RANDOM-EDGE has completed, whereas RANDOM-VERTEX will continue for one more step.

FIGURE 2. A visualization of the chains corresponding to RANDOM-VERTEX and RANDOM-EDGE on an instance with five jobs and five days. Each job has two feasible days, and only one job can be scheduled for each day. In each step, an unscheduled job is assigned to one remaining feasible day (solid green line), and the set of other jobs that were feasible for this day is revealed (dashed red lines). The numbers above each job denote the number of remaining feasible days. Under RANDOM-VERTEX, jobs are selected in uniformly at random, whereas under RANDOM-EDGE, they are selected in proportion to the number of remaining feasible days.

Lemma 1. *There exists a coupling of* $(\mathcal{M}_t^V, \mathcal{E}_t^V)$ *and* $(\mathcal{M}_t^E, \mathcal{E}_t^E)$ *such that for all* $t \leq \tau^E$ *and all* $k \in \{1, \dots, |\mathcal{D}|\}$ *the following hold:*

$$I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E). \tag{12}$$

$$\sum_{i \geq k} F_i(\mathcal{E}_t^V) \geq \sum_{i \geq k} F_i(\mathcal{E}_t^E). \tag{13}$$

Proof. Proof. First, (7) implies that if $I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E)$, then regardless of the realizations j_{t+1}^V, j_{t+1}^E, the chains can be coupled so that $d_{t+1}^V = d_{t+1}^E$, and therefore (6) implies $I(\mathcal{M}_{t+1}^V) = I(\mathcal{M}_{t+1}^E)$. In light of this fact, we write d_{t+1} in place of $d_{t+1}^V = d_{t+1}^E$, and $I(\mathcal{M}_t)$ in place of $I(\mathcal{M}_t^V) = I(\mathcal{M}_t^E)$.

We now turn to (13). For $X \in \{V, E\}$ we define

$$\mathcal{A}_{t+1}^X = \{(j_{t+1}^X, d) : d \in D_{t+1}^X\} \tag{14}$$

Lemma 2 states that it is possible to couple $\mathcal{A}_{t+1}^V, \mathcal{A}_{t+1}^E$ such that

$$\sum_{i \geq k} F_i(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq \sum_{i \geq k} F_i(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^V). \tag{15}$$

That is, an analog of (13) holds after stages V3 and E3. If $I_{d_{t+1}}(\mathcal{M}_{t+1}) > 0$, then by (8) and (9) $\mathcal{B}_{t+1}^V = \mathcal{B}_{t+1}^E = \emptyset$, so for $X \in \{V, E\}$, (10) implies that $\mathcal{E}_{t+1}^X = \mathcal{E}_t^X \cup \mathcal{A}_{t+1}^X$, and there is nothing more to do.

Otherwise, we must couple B_{t+1}^V and B_{t+1}^E. It follows from (15) that there exists a permutation ϕ on $\mathcal{J}$ such that $\phi(j_{t+1}^V) = j_{t+1}^E$, and for all j,

$$N_j(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq N_{\phi(j)}(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E). \tag{16}$$

For each j, if equality holds in (16), couple $B_{(t+1)j}^V$ and $B_{(t+1)\phi(j)}^E$ such that $B_{(t+1)j}^V = B_{(t+1)\phi(j)}^E$. If the inequality in (16) is strict, generate $B_{(t+1)j}^V$ and $B_{(t+1)\phi(j)}^E$ independently. This ensures that

$$N_j(\mathcal{E}_{t+1}^V) = N_j(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V \cup \mathcal{B}_{t+1}^V) \geq N_{\phi(j)}(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E \cup \mathcal{B}_{t+1}^E) = N_{\phi(j)}(\mathcal{E}_{t+1}^E) \tag{17}$$

for all j, and therefore that (13) holds. □

Lemma 2. *If* (13) *holds, then it is possible to couple* $\mathcal{A}_{t+1}^V, \mathcal{A}_{t+1}^E$ *such that for* $k \in \{1, \dots, |\mathcal{D}|\}$,

$$\sum_{i \geq k} F_i(\mathcal{E}_t^V \cup \mathcal{A}_{t+1}^V) \geq \sum_{i \geq k} F_i(\mathcal{E}_t^E \cup \mathcal{A}_{t+1}^E).$$

Proof. Proof. For $X \in \{V, E\}$ define $K_{t+1}^X = N_{j_{t+1}}(\mathcal{E}_t^X)$ to be the number of feasible days for job j_{t+1} that still have idle capacity. Then the definition of $\mathcal{A}_{t+1}^X$ in (5) implies that for $k \in \{1, \dots, |\mathcal{D}|\}$,

$$\sum_{i \geq k} F_i(\mathcal{E}_t^X \cup \mathcal{A}_{t+1}^X) = \sum_{i \geq k} F_i(\mathcal{E}_t^X) - \mathbf{1}(K_{t+1}^X \geq k).$$

Thus, it is enough to show that if $\sum_{i \geq k} F_i(\mathcal{E}_t^V) = \sum_{i \geq k} F_i(\mathcal{E}_t^E)$, it is possible to couple K_{t+1}^V and K_{t+1}^E such that $K_{t+1}^V \geq k$ implies that $K_{t+1}^E \geq k$. In other words, we must show that

$$\mathbb{P}(K_{t+1}^E \geq k | \mathcal{M}_t^E, \mathcal{E}_t^E) \geq \mathbb{P}(K_{t+1}^V \geq k | \mathcal{M}_t^V, \mathcal{E}_t^V).$$

Note that (4) implies that for $i \geq 1$,

$$\mathbb{P}(K_{t+1}^V = i | \mathcal{M}_t^V, \mathcal{E}_t^V) = \frac{F_i(\mathcal{E}_t^V)}{\sum_{j \geq 1} F_j(\mathcal{E}_t^V)}.$$

Analogously, (11) implies that

$$\mathbb{P}(K_{t+1}^E = i|\mathcal{M}_t^E, \mathcal{E}_t^E) = \frac{iF_i(\mathcal{E}_t^E)}{\sum_{j\geq 1} jF_j(\mathcal{E}_t^E)}.$$

It follows that if $\sum_{i\geq k} F_i(\mathcal{E}_t^V) = \sum_{i\geq k} F_i(\mathcal{E}_t^E)$, then

$$\begin{aligned}\mathbb{P}(K_{t+1}^V \geq k|\mathcal{M}_t^V, \mathcal{E}_t^V) &= \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^V)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^V)}\\ &\leq \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^V)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)}\\ &= \frac{\sum_{i\geq k} F_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)}\\ &\leq \frac{\sum_{i\geq k} iF_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} iF_i(\mathcal{E}_t^E)}\\ &= \mathbb{P}(K_{t+1}^E \geq k|\mathcal{M}_t^E, \mathcal{E}_t^E).\end{aligned}$$

The first inequality follows from (13), and the final inequality follows because

$$\begin{aligned}\frac{\sum_{i\geq k} F_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} F_i(\mathcal{E}_t^E)} &= \left(1 + \sum_{i<k} kF_i(\mathcal{E}_t^E)/\sum_{i\geq k} kF_i(\mathcal{E}_t^E)\right)^{-1}\\ &\leq \left(1 + \sum_{i<k} iF_i(\mathcal{E}_t^E)/\sum_{i\geq k} iF_i(\mathcal{E}_t^E)\right)^{-1} = \frac{\sum_{i\geq k} iF_i(\mathcal{E}_t^E)}{\sum_{i\geq 1} iF_i(\mathcal{E}_t^E)}.\end{aligned}$$

The inequality follows because replacing k with i makes the numerator smaller and the denominator bigger, and therefore decreases the quantity in parentheses (which is then inverted). $\square$

3. Proof of Theorem 2

The proof of Theorem 2 relies on coupling Markov chains corresponding to RANDOM-VERTEX and RANKING. However, the chain for RANDOM-VERTEX is different from the chain used to prove Theorem 1. Therefore, we begin by describing both Markov chains in Section 3.1, and then show how they can be coupled to prove the result in Section 3.2.

3.1. **Proof Step 1: Markov Chain Descriptions.**

We begin by defining Markov chains which describe the matching procedures RANDOM-VERTEX and RANKING. In both chains, we fix the order $\succ^{\mathcal{J}}$, and label jobs so that $j_1 \succ^{\mathcal{J}} j_2 \succ^{\mathcal{J}} \cdots \succ^{\mathcal{J}} j_{|\mathcal{J}|}$. At each step t, the chains will attempt to match job j_t.

Given any matching $\mathcal{M}$ and $d \in \mathcal{D}$, let $I_d(\mathcal{M})$ be the idle capacity of day d as defined in (1), and define

$$\mathcal{D}_i(\mathcal{M}) = \{d : I_d(\mathcal{M}) = i\}, \qquad D_i(\mathcal{M}) = |\mathcal{D}_i(\mathcal{M})|. \tag{18}$$

$$\mathcal{D}_+(\mathcal{M}) = \mathcal{D}\backslash\mathcal{D}_0(\mathcal{M}), \qquad D_+(\mathcal{M}) = |\mathcal{D}_+(\mathcal{M})|. \tag{19}$$

We refer to days with no idle capacity (days in $\mathcal{D}_0(\mathcal{M})$) as "unavailable," and days with idle capacity (days in $\mathcal{D}_+(\mathcal{M})$) as "available." We say that day d is "feasible" for job j if $(j, d) \in \mathcal{E}$.

We start with an empty matching, and attempt to match j_1. Suppose that the order $\succ_{j_1}^{\mathcal{D}}$ in which j_1 considers days is known, but feasible days for j_1 have not yet been revealed. If j_1 has N

feasible days, it j_1 will choose the k^{th} ranked day (according to $\succ_{j_1}^{\mathcal{D}}$) with probability

$$f(k,N) = \binom{|\mathcal{D}|-k}{N-1} / \binom{|\mathcal{D}|}{N}. \tag{20}$$

This is because the numerator counts the number of subsets of $\mathcal{D}$ of size N that include the k^{th}-ranked day and no higher-ranked day. Note that when $N = 1$, all days are equally likely, but as N grows, higher-ranked days become increasingly likely. At the extreme where $N = |\mathcal{D}|$, the highest-ranked day is chosen with certainty.

In general, some days may no longer be available at step t. Thus, the probability that job j_t matches to a particular day depends not only on the ranking over days $\succ_{j_t}^{\mathcal{D}}$ but also on the set of available days: day d is selected whenever it is the highest-ranked feasible and *available* day. For any order over days $\succ$, any subset of days $\mathcal{D}' \subseteq \mathcal{D}$, and any day $d \in \mathcal{D}'$, define

$$\text{Rank}(d \mid \mathcal{D}', \succ) = 1 + |\{d' \in \mathcal{D}' : d' \succ d\}. \tag{21}$$

to be the ranking of d among days in $\mathcal{D}'$. In step t, each day $d \in \mathcal{D}_+(\mathcal{M}_{t-1})$ has effectively been "promoted": the probability that job j_t matches to d given that $\mathcal{M}_{t-1} = \mathcal{M}$, and $\succ_{j_t}^{\mathcal{D}} = \succ$ is

$$q_t(d \mid \mathcal{M}, \succ) = \begin{cases} 0 & : d \in \mathcal{D}_0(\mathcal{M}). \\ f(\text{Rank}(d \mid \mathcal{D}_+(\mathcal{M}), \succ), \bar{N}_{j_t}) & : d \in \mathcal{D}_+(\mathcal{M}). \end{cases} \tag{22}$$

We also define

$$\begin{aligned} q_t(\emptyset \mid \mathcal{M}, \succ) &= 1 - \sum_{d \in \mathcal{D}} q_t(d \mid \mathcal{M}, \succ) \\ &= \binom{D_0(\mathcal{M})}{\bar{N}_{j_t}} / \binom{|\mathcal{D}|}{\bar{N}_{j_t}}. \end{aligned} \tag{23}$$

to be the probability that job j_t goes unassigned. Note that the second equality follows because j_t goes unassigned if and only if all of its feasible days are unavailable. Because the probability of this does not depend on the order $\succ$, we sometimes write $q_t(\emptyset \mid \mathcal{M})$ in place of $q_t(\emptyset \mid \mathcal{M}, \succ)$.

From these expressions, there is a natural Markov chain description of both procedures. Start from an empty matching, and at each step t, reveal the order in which job j_t will consider days and the feasible days for j_t, and use these to determine the day (if any) to which j_t is matched. Instead, we will define chains that reveal *only* the match outcome for j_t. As a result, the state (and history) are fully tracked by the matching $\mathcal{M}_t$ consisting of the matches after jobs $j_1, \ldots, j_t$ have been considered.

3.1.1. *Markov Chain Description of* RANDOM-VERTEX.

Let $\mathcal{O}$ be the set of orderings of days. Under RANDOM-VERTEX, the orders $\succ_j^{\mathcal{D}}$ are drawn uniformly at random from $\mathcal{O}$. Therefore, the probability that day d is assigned at step t given $\mathcal{M}_{t-1} = \mathcal{M}$ is

$$p_t^V(d \mid \mathcal{M}) = \frac{1}{|\mathcal{D}|!} \sum_{\succ \in \mathcal{O}} q_t(d \mid \mathcal{M}, \succ). \tag{24}$$

By symmetry, each available day is equally likely to match next. Therefore, the matching process for RANDOM-VERTEX can be described by the following Markov chain. Start with $\mathcal{M}_0^V = \emptyset$, and at step t, do the following:

V1. With probability $q_t(\emptyset \mid \mathcal{M}_t^V)$, job j_t goes unassigned: $\mathcal{M}_t^V = \mathcal{M}_{t-1}^V$.

V2. Otherwise, match j_t to a uniformly random day $d_t \in \mathcal{D}_+(\mathcal{M}_{t-1}^V)$: $\mathcal{M}_t^V = \mathcal{M}_{t-1}^V \cup (j_t, d_t)$.

It follows from this description that equation in (24) can be re-expressed as

$$p_t^V(d \mid \mathcal{M}) = \begin{cases} 0 & : d \in \mathcal{D}_0(\mathcal{M}). \\ \frac{1-q_t(\emptyset \mid \mathcal{M})}{D_+(\mathcal{M})} & : d \in \mathcal{D}_+(\mathcal{M}). \end{cases} \tag{25}$$

3.1.2. *Markov Chain Description of* RANKING.

As mentioned above, our Markov chain will not reveal the feasible days for each job or the universal order $\succ^{\mathcal{D}}$ over days. Instead, it reveals only which day (if any) each job is matched to. From this, we must make inferences about $\succ^{\mathcal{D}}$, and through this, about the probability that each day will be selected at the next step.

Given a matching $\mathcal{M}$, let $d_t(\mathcal{M})$ be the day to which job j_t is assigned in $\mathcal{M}$ (with $d_t = \emptyset$ if j_t is unassigned), and let $\mathcal{M}_t$ denote the subset of the matching involving jobs $j_1, \ldots, j_t$. For any order over days $\succ$, the likelihood of $\mathcal{M}_t$ given that all jobs consider days in the order $\succ$ is

$$\mathbb{P}_t(\mathcal{M} \mid \succ) = \prod_{s=1}^{t} q_s(d_s(\mathcal{M}) \mid \mathcal{M}_{s-1}, \succ). \tag{26}$$

Because $\succ^{\mathcal{D}}$ is drawn uniformly at random from the set of order over days $\mathcal{O}$, Bayes' theorem implies that for any $\succ \in \mathcal{O}$, the conditional probability that $\succ^{\mathcal{D}}$ equals $\succ$ given $\mathcal{M}_t = \mathcal{M}$ is

$$\mathbb{P}_t(\succ \mid \mathcal{M}) = \frac{\mathbb{P}_t(\mathcal{M} \mid \succ)}{\sum_{\succ' \in \mathcal{O}} \mathbb{P}_t(\mathcal{M} \mid \succ')}. \tag{27}$$

From this, we can compute the conditional probability that job j is assigned to day d, given $\mathcal{M}_{t-1} = \mathcal{M}$:

$$p_t^R(d \mid \mathcal{M}) = \sum_{\succ \in \mathcal{O}} \mathbb{P}_{t-1}(\succ \mid \mathcal{M}) q_t(d \mid \mathcal{M}, \succ). \tag{28}$$

Putting everything together, we have the following Markov chain description of RANKING. Start with $\mathcal{M}_0^R = \emptyset$, and at step t, do the following.

R1. For each $d \in \mathcal{D}_+(\mathcal{M}_{t-1}^R)$, with probability $p_t^R(d \mid \mathcal{M}_{t-1}^R)$, match j_t to d: $\mathcal{M}_t^R = \mathcal{M}_{t-1}^R \cup (j_t, d)$.

R2. Otherwise job j_t goes unassigned: $\mathcal{M}_t^R = \mathcal{M}_{t-1}^R$.

Note that the conditional probability that job j_t goes unassigned is $q_t(\emptyset \mid \mathcal{M}_{t-1}^R)$.

3.2. **Proof Step 2: Coupling.** The claim when $C_d = 1$ for all d follows almost immediately from the Markov chain descriptions. At each step t, (23) says that the probability that j_t matches depends only on the number of unavailable days, and not on the order $\succ$. Thus, so long as the number of unavailable days is equal for the two chains after step $t-1$, they can be coupled so that j_t matches in $\mathcal{M}_t^V$ if and only if it matches in $\mathcal{M}_t^R$. Because $C_d = 1$ for all days, the number of unavailable days is equal to the number of matches, and thus it remains equal for the two chains after step t.

We now turn to the claim when $C_d = 2$ for all d, and show that under RANKING, days which have one assigned job are weakly more likely to be selected next than days which have no assigned jobs. The intuition is that having been previously selected is a positive signal about d's position in the ranking $\succ^{\mathcal{D}}$.

Lemma 3. *Suppose that $C_d = 2$ for all d. If $d_1 \in \mathcal{D}_1(\mathcal{M}_{t-1}^R)$ and $d_2 \in \mathcal{D}_2(\mathcal{M}_{t-1}^R)$, then*

$$p_t^R(d_2 \mid \mathcal{M}_{t-1}^R) \le p_t^R(d_1 \mid \mathcal{M}_{t-1}^R).$$

As a result,

$$\sum_{d \in \mathcal{D}_2(\mathcal{M}_{t-1}^R)} p_t^R(d \mid \mathcal{M}_{t-1}^R) \le \sum_{d \in \mathcal{D}_2(\mathcal{M}_{t-1}^R)} p_t^V(d \mid \mathcal{M}_{t-1}^R).$$

The second part of the lemma says that the chance that j_{t+1} is assigned to a day with no assigned job is lower than it would be if j_t were to draw a new random order over days (rather than using the universal order $\succ^{\mathcal{D}}$).

Proof. Throughout this proof, we fix t, and the matching $\mathcal{M}_{t-1}^R$, and remove explicit reference to these objects in order to reduce notational clutter. Thus, (28) will be written as

$$p^R(d) = \sum_{\succ} \mathbb{P}(\succ) q(d \mid \succ).$$

Given any order over days $\succ$, let $\succ'$ denote the order which is identical to $\succ$ except that d_1 and d_2 are transposed. Then we can express the difference $p^R(d_1) - p^R(d_2)$ using the following sum over orders in which d_1 precedes d_2:

$$p^R(d_1) - p^R(d_2) = \sum_{\{\succ : d_1 \succ d_2\}} \mathbb{P}(\succ)\left(q(d_1 \mid \succ) - q(d_2 \mid \succ)\right) + \mathbb{P}(\succ')\left(q(d_1 \mid \succ') - q(d_2 \mid \succ')\right). \tag{29}$$

Note that (21) and (22) imply that $q(d| \succ)$ depends only on d's rank (according to $\succ$) among available days. Therefore, exchanging the positions of d_1 and d_2 exchanges their probability of being selected: $q(d_2 \mid \succ) = q(d_1 \mid \succ')$, and $q(d_2 \mid \succ') = q(d_1 \mid \succ)$. We can substitute these equalities into (29) to get

$$p^R(d_1) - p^R(d_2) = \sum_{\{\succ : d_1 \succ d_2\}} \left(\mathbb{P}(\succ) - \mathbb{P}(\succ')\right)\left(q(d_1 \mid \succ) - q(d_1 \mid \succ')\right). \tag{30}$$

We claim that for every $\succ$ such that $d_1 \succ d_2$,

$$q(d_1 \mid \succ) - q(d_1 \mid \succ') \geq 0 \tag{31}$$

$$\mathbb{P}(\succ) \quad - \quad \mathbb{P}(\succ') \quad \geq 0. \tag{32}$$

From this, (30) implies the first claim of Lemma 3.

Both (31) and (32) follow from (20), (21) and (22), which jointly imply that d_1 is more likely to be selected under the order $\succ$ in which it appears earlier. This directly implies (31). To see that it also implies (32), note that by (27), (32) is equivalent to

$$\mathbb{P}(\mathcal{M}_{t-1}^R \mid \succ) \geq \mathbb{P}(\mathcal{M}_{t-1}^R \mid \succ').$$

By (26), each of these likelihoods can be expressed as a product of $t-1$ terms, and the only term that differs is the one corresponding to the step at which d_1 was selected. But (20), (21) and (22) imply that this term is (weakly) larger for $\succ$ than $\succ'$. Therefore (32) holds, completing the proof of the first claim of Lemma 3.

We now turn to the second claim. The argument is fairly simple, so we present it in English, rather than symbolically. Note that $p^R(d)$ gives the chance that j_t matches to d when it uses the order $\succ^{\mathcal{D}}$ used by all previous jobs, while $p^V(d)$ gives the chance that j_t matches to d when it uses a uniformly random order. In this case, j_t is equally likely to match to each available day. When using the order $\succ^{\mathcal{D}}$, the first part of the Lemma establishes that days in $\mathcal{D}_2$ are the least likely to be matched. Because the overall probability that j_t matches (that is, the probability of matching to a day in $\mathcal{D}_1$ or $\mathcal{D}_2$) does not depend on the order in which j_t considers days, it follows that j_t is less likely to match to a day in $\mathcal{D}_2$ when using the order $\succ^{\mathcal{D}}$. □

Next, we use Lemma 3 to demonstrate that the chains $\mathcal{M}^R$ and $\mathcal{M}^V$ can be coupled such that under $\mathcal{M}^R$ there are more days with two unfilled positions and more unfilled positions overall.

Lemma 4. *Suppose that $C_d = 2$ for all d. There exists a coupling of the chains $\mathcal{M}^R$ and $\mathcal{M}^V$ such that at each step t,*

$$D_2(\mathcal{M}_t^R) \geq D_2(\mathcal{M}_t^V) \tag{33}$$

$$2D_2(\mathcal{M}_t^R) + D_1(\mathcal{M}_t^R) \geq 2D_2(\mathcal{M}_t^V) + D_1(\mathcal{M}_t^V). \tag{34}$$

Note that for any matching $\mathcal{M}$, $|\mathcal{M}| + D_1(\mathcal{M}) + 2D_2(\mathcal{M}) = \sum_d C_d = 2|\mathcal{D}|$, so (34) is equivalent to $|\mathcal{M}_t^V| \geq |\mathcal{M}_t^R|$. In other words, Lemma 4 immediately implies the $C_d = 2$ case of Theorem 2.

Proof. The proof is by induction. Both inequalities trivially hold at $t = 0$, so we assume that (33) and (34) holds for $\mathcal{M}^R_{t-1}$ and $\mathcal{M}^V_{t-1}$, and show that the chains can be coupled such that this continues to hold at step t. To simplify notation, we write D^R_i and D^V_i in place of $D_i(\mathcal{M}^R_{t-1})$ and $D_i(\mathcal{M}^V_{t-1})$, and $\mathcal{D}^R_i$ and $\mathcal{D}^V_i$ in place of $\mathcal{D}_i(\mathcal{M}^R_{t-1})$ and $\mathcal{D}_i(\mathcal{M}^V_{t-1})$.

Notice that the expressions on both sides of (33) and (34) are integer-valued, and can only stay the same or decrease by one when moving from $\mathcal{M}_{t-1}$ to $\mathcal{M}_t$. Therefore, if either inequality is strict at time $t-1$, it will continue to hold at time t. This leaves three possibilities to consider:

- Both (33) and (34) hold with equality.
- Equation (33) holds strictly but (34) holds with equality.
- Equation (34) holds strictly but (33) holds with equality.

If both (33) and (34) hold with equality, then $D^R_i = D^V_i$ for $i \in \{0, 1, 2\}$. It follows from (23) that the chance that j_t matches is equal under the two chains, so the chains can be coupled so that a match forms in $\mathcal{M}^R$ whenever one forms in $\mathcal{M}^V$, ensuring that (34) continues to hold with equality. Furthermore, Lemma 3 implies that the chance that j_t is assigned to a day in $\mathcal{D}_2$ is smaller under $\mathcal{M}^R$ than $\mathcal{M}^V$, so the chains can be coupled such that (33) continues to hold.

If equation (33) holds strictly but (34) holds with equality, it follows that $D^R_2 + D^R_1 < D^V_2 + D^V_1$, and therefore that $D^R_0 > D^V_0$. That is, there are more unavailable days in chain $\mathcal{M}^R$. By (23), this implies that job j_t is weakly more likely to match in $\mathcal{M}^V$ than $\mathcal{M}^R$. Therefore, we can couple the chains such that whenever j_t matches in $\mathcal{M}^R$, it also matches in $\mathcal{M}^V$, ensuring that (34) continues to hold at step t.

The trickiest case is when (33) holds with equality and (34) holds strictly. In order to ensure that (33) continues to hold at step t, we must show that we are more likely to assign j_t to a day with two remaining slots under RANDOM-VERTEX – that is,

$$p^R_t(\mathcal{D}^R_2|\mathcal{M}^R_{t-1}) \le p^V_t(\mathcal{D}^V_2|\mathcal{M}^V_{t-1}),$$

where for $X \in \{V, R\}$ and $\mathcal{D}' \subseteq \mathcal{D}$, we use $p^X_t(\mathcal{D}' \mid \mathcal{M})$ as shorthand for $\sum_{d\in\mathcal{D}'} p^X_t(d \mid \mathcal{M})$. Lemma 3 states that

$$p^R_t(\mathcal{D}^R_2 \mid \mathcal{M}^R_{t-1}) \le p^V_t(\mathcal{D}^R_2 \mid \mathcal{M}^R_{t-1}).$$

Therefore, it suffices to show that

$$p^V_t(\mathcal{D}^R_2 \mid \mathcal{M}^R_{t-1}) \le p^V_t(\mathcal{D}^V_2|\mathcal{M}^V_{t-1}). \tag{35}$$

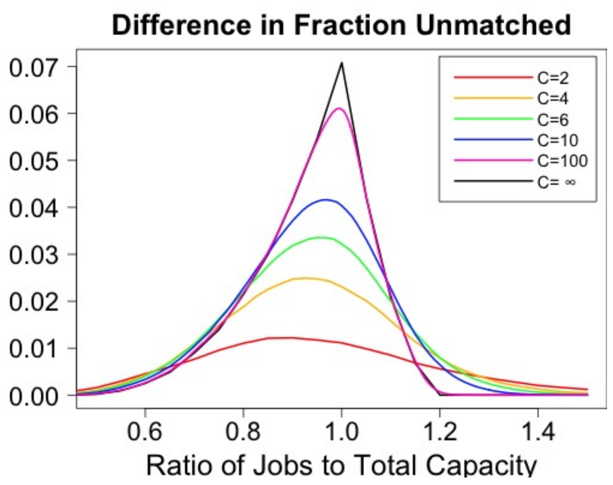
By (24), the left side of this inequality gives the probability that j_t matches to a day that is unmatched in $\mathcal{M}^R_{t-1}$, if j_t considers days in a *uniformly random order* (rather than the order $\succ^{\mathcal{D}}$). Meanwhile, the right side is the probability that j_t matches to a day that is unmatched in $\mathcal{M}^V_{t-1}$ (again when j_t considers days in a uniformly random order).

From (25) it follows that for any $d \in \mathcal{D}_+(\mathcal{M})$,

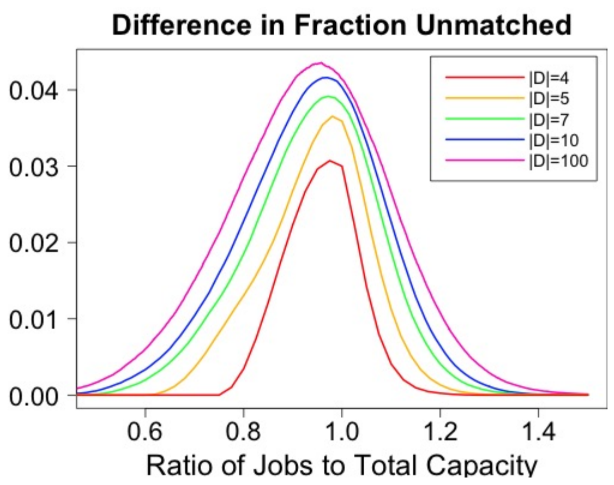
$$p^V_t(\mathcal{D}_2(\mathcal{M}) \mid \mathcal{M}) = D_2(\mathcal{M}) p^V_t(d \mid \mathcal{M}).$$

By (25) and (23), $p^V_t(d \mid \mathcal{M})$ depends on $\mathcal{M}$ only through $D_0(\mathcal{M})$. By (24), $p^V_t(d \mid \mathcal{M})$ is increasing in $D_0(\mathcal{M})$: for any order $\succ$, increasing the number of unavailable days weakly improves d's effective rank. Thus, to establish (35), it suffices to show that $D^R_2 = D^V_2$ and $D^R_0 < D^V_0$. These follow immediately from the assumption that (33) holds with equality and (34) holds strictly. □

We now briefly discuss why our proof applies only when days have identical capacity $C \le 2$. The intuition underlying Lemma 3 is that days which have been selected more often in the past should be more likely to be selected in the future. While this should continue to hold "on average" when $C > 2$, it no longer can be made to hold along each sample path. The reason is that although being selected is always a positive signal about a day's position in $\succ^{\mathcal{D}}$, the *strength* of this signal depends on the number of other feasible and available days. To illustrate, consider a simple case with only two days. Both days are feasible for the first job, which chooses d_1. This reveals that $d_1 \succ^{\mathcal{D}} d_2$.

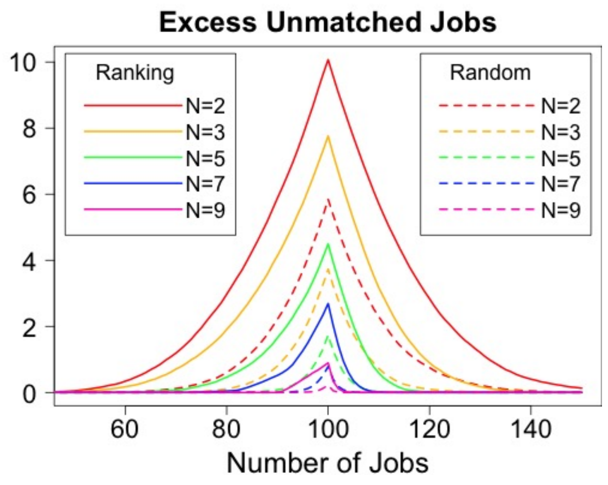


(A) Each job has 3 feasible days, there are $|\mathcal{D}| = 10$ days, and the capacity of each day varies. For a balanced market, the difference between RANKING and RANDOM-VERTEX increases with C.

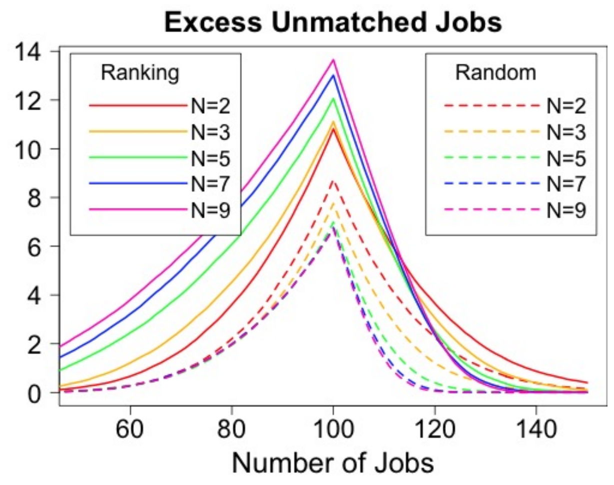


(B) Each job has 3 feasible days, each day has capacity $C = 10$, and the number of days varies. The difference between RANKING and RANDOM-VERTEX increases with $|\mathcal{D}|$.

Figure 3. Simulations comparing unmatched jobs under RANKING and RANDOM-VERTEX when all days have identical capacity C, for various $|\mathcal{J}|, |\mathcal{D}|$, and C. The x-axis shows the ratio of jobs to total capacity $|\mathcal{J}|/(C \times |\mathcal{D}|)$, while the y-axis gives the difference in the fraction of jobs that are unmatched $(|\mathcal{M}^V| - |\mathcal{M}^R|)/|\mathcal{J}|$. Theorem 2 proves that when $C = 2$, RANDOM-VERTEX matches more jobs than RANKING. The simulations show that this also holds when $C > 2$.



(A) All jobs have N feasible days. Increasing N decreases unmatched jobs under both RANKING and RANDOM-VERTEX.



(B) Half of jobs have N feasible days, the rest have 1 feasible day. Increasing N decreases unmatched jobs under RANDOM-VERTEX, but often <u>increases</u> unmatched jobs under RANKING.

Figure 4. Simulations when $|\mathcal{D}| = 10$ and all days have capacity $C = 10$. The y-axis shows the difference between the number of unmatched jobs and the minimum possible number of unmatched jobs $\max(|\mathcal{J}| - C|\mathcal{D}|, 0)$. Differences between RANDOM-VERTEX and RANKING are biggest when the number of feasible days varies significantly across jobs.

Suppose that the next two jobs have only one feasible day, d_2. After these jobs have matched, d_2 has less idle capacity than d_1, but d_1 is nevertheless more likely to be matched in the next step. In other words, the natural generalization of Lemma 3 does not hold.

This illustrates a limitation of our proof technique, rather than an example in which RANKING matches more jobs. We conjecture that RANDOM-VERTEX continues to match more jobs than RANKING when $C > 2$. Figures 3 and 4 present simulation results supporting this conjecture.

4. Discussion

The greedy matching algorithms RANDOM-EDGE, RANDOM-VERTEX and RANKING have primarily been compared according to their worst-case performance for one-to-one matching. This paper complements prior work by providing an average-case analysis for one-to-one and many-to-one matching. In some cases, such as the comparison between RANDOM-EDGE and RANDOM-VERTEX in Theorem 1, average-case analysis confirms the story told by worst-case guarantees. In others, it does not: while RANKING offers a better worst-case guarantee than RANDOM-VERTEX for one-to-one matching, they have identical average-case performance. Furthermore, Theorem 2 establishes that RANDOM-VERTEX outperforms RANKING for two-to-one matching.

Our results also have implications for weighted matching. If edge weights are drawn iid, then RANDOM-EDGE corresponds to the natural greedy procedure which sequentially adds the highest-weight edge that maintains feasibility. This guarantees a matching with weight at least half of optimal. Meanwhile, RANDOM-VERTEX corresponds to a vertex-iterative procedure that considers jobs in random order and selects the highest-weight feasible edge. Although this approach does not offer any worst-case guarantee, Theorem 1 implies that if the distribution of edge weights is sufficiently concentrated, it will on average produce a higher-weight matching than greedy edge selection.

Another example where worst-case and average-case analysis may disagree is the case of the MINGREEDY algorithm, which considers jobs in increasing order of their degree. Although this algorithm guarantees only the trivial factor of 1/2 (Besser and Poloczek 2017), we conjecture that on average, it outperforms RANDOM-VERTEX for any degree distribution among jobs.

Our results are primarily intended to contribute to the academic literature, rather than to offer immediate policy guidance. However, we briefly note that variants of the greedy algorithms studied in this paper arise frequently in practice. In many contexts, for example, "jobs" arrive dynamically and must be assigned immediately. The most well-studied example is online advertising, where impressions are assigned to advertisers in real time: see for example Mehta et al. (2007), Goel and Mehta (2008), Feldman et al. (2009), Aggarwal et al. (2011), and a helpful survey by Mehta (2013). As another example, many wilderness areas have a limited number of camping permits for each day, which are assigned dynamically: on a first-come-first-served basis, prospective campers can select among days for which permits remain. This system corresponds to a vertex-iterative algorithm in which $\succ^{\mathcal{J}}$ encodes campers' order of arrival, and $\succ_j^{\mathcal{D}}$ encodes the preferences of camper j.

Even when items are allocated in a single round, it is common to use a "random serial dictatorship," which is a vertex-iterative algorithm where the order $\succ^{\mathcal{J}}$ is selected uniformly at random.[7] In addition to its procedural fairness, this approach is simple to describe and implement, and incentivizes applicants to submit their preferences and constraints truthfully. By contrast, algorithms which select maximum matchings have none of these features. The algorithms RANDOM-VERTEX and RANKING can be thought of as a random serial dictatorship in which applicants have independent or identical preferences over days, respectively. Meanwhile, the procedure RANDOM-EDGE arises if a new lottery number is assigned to each *application* rather than each *applicant*.

[7]This is in fact how permits for the popular Half Dome hike in Yosemite are allocated – see `https://www.nps.gov/yose/planyourvisit/hdpermits.htm`.

To get closer to these applications, bells and whistles can be added to our basic model. For example, Appendix A extends the conclusions of Theorem 1 to settings where jobs have different "sizes" and some days are more likely to be feasible than others.

Although our work focuses on directional comparisons, its practical relevance also depends on the *magnitude* of the differences between procedures. This of course depends on the graph parameters. For example, differences will be small if all jobs have sufficiently high degree, or if there is a large imbalance between the sides. Meanwhile, Figure 3 suggests that in balanced markets, the difference between RANDOM-VERTEX and RANKING is largest in large markets when days have large capacities. Figure 4 suggests that the difference is largest when there is a lot of heterogeneity in the number of feasible days across jobs. In particular, it shows that increasing the number of feasible days for some jobs may *decrease* the expected number of jobs matched by RANKING.

Asymptotic analysis may help to explore these phenomena. There are (at least) two ways to let the market grow large: by increasing the number of days, or the capacity of each day. In the former limit, the differential equation method from Wormald (1995, 1999) makes it possible to derive asymptotic expressions for the number of matches produced by RANDOM-VERTEX and RANDOM-EDGE. Appendix B.1 provides such expressions for sparse bipartite Erdös-Renyi random graphs. These expressions suggest that the difference between RANDOM-VERTEX and RANDOM-EDGE is fairly small. Meanwhile, Appendix B.2 uses the latter limit to derive asymptotic expressions for the number of matches produced by RANDOM-VERTEX and RANKING. We leave a more complete analysis of these expressions for future work.

Appendix A. Extensions of Theorem 1

This appendix extends the conclusions of Theorem 1 to more general settings.

A.1. **Job Sizes.**

In our first extension, jobs have different "sizes" S_j. For example, C_d might represent the number of hours that developer d can work, and S_j the number of hours that job j will take to complete. Alternately, in the context of hiking permits or theater performances, C_d might represent the number of permits or tickets available on day d, and S_j the number of permits or tickets requested by group j.

We add to the primitives $\bar{\mathbf{N}}$ and $\mathbf{C}$ the size vector $\mathbf{S} \in \mathbb{N}^{|\mathcal{J}|}$. We make two key assumptions:

- Size is independent from the number of feasible dates. Formally, the size of job j is $S_{\rho(j)}$, where ρ is a uniform random bijection from $\mathcal{J}$ to $\{1, \ldots, |\mathcal{J}|\}$.
- Small amounts of overbooking are allowed. Formally, we consider it feasible to add (j, d) to $\mathcal{M}$ so long as $(j, d) \in \mathcal{E}$ and d has extra capacity under $\mathcal{M}$ (even if $S_{\rho(j)}$ exceeds this excess capacity).

Under these assumptions, an analogous proof holds. In particular, we reveal ρ incrementally: after step t, we have a matching $\mathcal{M}_t$ and an injection $\rho_t : \{j : (j, d) \in \mathcal{M}_t \text{ for some } d\} \to \{1, \ldots, |\mathcal{D}|\}$. In steps V1 and E1, we reveal the value $\rho_{t+1}(j_{t+1})$ (that is, the size of the job matched at time $t + 1$) and ensure that ρ_{t+1} is consistent with ρ_t on the domain of ρ_t. We define

$$I_d(\mathcal{M}_t, \rho_t) = C_d - \sum_{j' : (j', d) \in \mathcal{M}_t} S_{\rho_t(j')},$$

and replace (8) with

$$\mathbb{P}(B_{(t+1)j} = 1 | \mathcal{E}_t, \mathcal{M}_t, j_{t+1}, \rho_{t+1}, D_{t+1}, d_{t+1}) = \frac{N_j(\mathcal{E}_t)}{|\{d : I_d(\mathcal{M}_t) > 0\}|} \mathbf{1}(j \neq j_{t+1}) \mathbf{1}(I_{d_{t+1}}(\mathcal{M}_{t+1}) \leq 0).$$

We then couple the two chains as in Lemma 1. In order to maintain (12), we ensure that for all $t < \tau^E$ we have $d^V_{t+1} = d^E_{t+1}$ (as before) and $\rho^V_{t+1}(j^V_{t+1}) = \rho^E_{t+1}(j^E_{t+1})$. That is, the size of the group matched at time t is identical in the two chains. It follows that both the number and total size of matched jobs is higher under RANDOM-VERTEX than RANDOM-EDGE.

A.2. **Non-Uniform Scheduling Constraints.**

In our second extension, some days are more likely than others to be feasible for each job. For example,a programming task may be more likely to be feasible for an experienced developer than an inexperienced one. Analogously, weekends may be feasible for more hikers or theater-goers than weekdays.

We add to the primitives $\bar{\mathbf{N}}$ and $\mathbf{C}$ (and, if desired, $\mathbf{S}$) the probability vector $\mathbf{P} \in [0, 1]^{\mathcal{D}}$ with $\sum_d P_d = 1$. We make the following key assumption:

- The $\bar{\mathbf{N}}_j$ edges involving job j are drawn independently (with replacement) from $\mathbf{P}$.

Formally, this requires allowing $\mathcal{E}$ to be a multiset – that is, (j, d) may appear in $\mathcal{E}$ multiple times. The stages V2-V4 and E2-E4 are modified as follows.

V2/E2 Determine the multiset $\mathcal{A}_{t+1} = \{(j_{t+1}, d_{(t+1)i})\}_{i=1}^{N_j(\mathcal{M}_t)}$ by drawing the $d_{(t+1)i}$ iid, with

$$\mathbb{P}(d_{(t+1)i} = d | \mathcal{M}_t, \mathcal{E}_t, j_{t+1}) = \frac{P_d \mathbf{1}(I_d(\mathcal{M}_t) > 0)}{\sum_{d'} P_{d'} \mathbf{1}(I_{d'}(\mathcal{M}_t) > 0)}.$$

V3/E3 Choose $e_{t+1} = (j_{t+1}, d_{t+1})$ uniformly from $\mathcal{A}_{t+1}$.

V4/E4 As before, if $j = j_{t+1}$ or $I_{d_{t+1}}(\mathcal{M}_{t+1}) > 0$, then $B_{(t+1)j} = 0$. Otherwise, the value $B_{(t+1)j}$ follows a binomial distribution with parameters $N_j(\mathcal{M}_t)$ and $P_{d_{t+1}} / \sum_{d'} P_{d'} \mathbf{1}(I_{d'}(\mathcal{M}_t) > 0)$. For $j \in \mathcal{J}$, let $\mathcal{B}_{t+1}$ contain $B_{(t+1)j}$ copies of the edge (j, d_{t+1}).

We then couple the two chains as in Lemma 1. The only necessary modification is to the coupling of $B^V_{(t+1)j}$ and $B^E_{(t+1)j}$ following (16). In this case, draw the values $B^E_{(t+1)j}$ as described in V4/E4 and set

$$B^V_{(t+1)j} = B^E_{(t+1)\phi(j)} + \tilde{B}_{(t+1)j},$$

where $\tilde{B}_{(t+1)j} = 0$ if $j = j^V_{t+1}$ or $I_{d^V_{t+1}}(\mathcal{M}^V_{t+1}) > 0$, and otherwise $\tilde{B}_{(t+1)j}$ follows a binomial distribution with parameters $N_j(\mathcal{E}^V_t) - N_{\phi(j)}(\mathcal{E}^E_t)$ and $P_{d_{t+1}}/\sum_{d'} P_{d'}\mathbf{1}(I_{d'}(\mathcal{M}_t) > 0)$. As before, this coupling ensures that

$$N_j(\mathcal{E}^V_t \cup \mathcal{A}^V_{t+1} \cup \mathcal{B}^V_{t+1}) \geq N_{\phi(j)}(\mathcal{E}^E_t \cup \mathcal{A}^E_{t+1} \cup \mathcal{B}^E_{t+1}).$$

Appendix B. Asymptotic Performance

In this appendix, we briefly study the magnitude of the differences identified in Theorems 1 and 2. We consider two different asymptotic regimes. The expressions in Appendix B.1 correspond to a case where the capacity of each day is $C_d = 1$, and the number of jobs and days both grow. The expressions in Appendix B.2 correspond to a case where the number of days is fixed, and the number of jobs and capacity of each day grow.

B.1. **Comparing RANDOM-VERTEX and RANDOM-EDGE in sparse Erdös-Renyi random graphs.**

Theorem 1 states that on average, RANDOM-VERTEX results in a larger matching than RANDOM-EDGE. However, it is silent about the magnitude of this difference. One way to address this question is to conduct asymptotic analysis as $|\mathcal{J}|$ and $|\mathcal{D}|$ grow, with $|\mathcal{J}|/|\mathcal{D}| = \rho$ fixed. Using techniques developed in Wormald (1995, 1999), we can write down a differential equation whose solution gives the expected match size under each algorithm.

This differential equation has a clean solution for one-to-one matching in sparse Erdös-Renyi random graphs. We obtain the following expressions for the fraction of jobs that are scheduled under RANDOM-VERTEX and RANDOM-EDGE, respectively, as a function of the imbalance ρ and the average number of feasible days for each job ℓ:

$$F_V = \frac{1}{\rho}\left(1 - \frac{1}{\ell}\log\left(1 + (e^{\ell} - 1)e^{-\rho\ell}\right)\right), \tag{36}$$

$$F_E = \begin{cases} \frac{e^{(\rho-1)\ell}-1}{\rho e^{(\rho-1)\ell}-1} & : \rho > 1 \\ \frac{\ell}{1+\ell} & : \rho = 1 \\ \frac{e^{(1-\rho)\ell}-1}{e^{(1-\rho)\ell}-\rho} & : \rho < 1. \end{cases} \tag{37}$$

Theorem 3 in Mastin and Jaillet (2018) gives the expression for F_V in the special case where $\rho = 1$. Their analysis directly implies (36) in the case $\rho < 1$. They do not give the expression in (36) for $\rho > 1$, although it would be possible to obtain with small modifications to their work. Dyer et al. (1993) study RANDOM-EDGE in non-bipartite Erdös-Renyi graphs, and obtain the $\ell/(\ell+1)$ expression that we claim for $\rho = 1$. We are unaware of any paper where the expression in (37) appears for general ρ.

For the case $\rho = 1$, these expressions imply that the fraction of unscheduled jobs is $\frac{1}{\ell+1}$ under RANDOM-EDGE and $\frac{\log(2-e^{-\ell})}{\ell} \geq \frac{\log(2)}{\ell+1}$ under RANDOM-EDGE. Both expressions are inversely proportional to ℓ, and within a factor of $\log(2) \approx 0.69$ of each other. This suggests that the gap between these procedures is small compared to the gap between them and a maximum matching (under which the number of unscheduled jobs decreases exponentially in ℓ). For general ρ, a numerical search suggests that in sparse Erdös-Renyi random graphs, the expected size of the matching produced by RANDOM-EDGE is always at least 96% of the expected size under RANDOM-VERTEX.

When we move beyond Erdös-Renyi graphs, the gap between these algorithms can be larger. For example, if there are equally many jobs and days, and half of jobs have degree 1 while the other half have degree 10, then RANDOM-EDGE expects to schedule approximately 71% of all jobs, whereas RANDOM-VERTEX schedules 78%; on this family, a maximum matching schedules approximately 89%.

B.2. **Comparing RANDOM-VERTEX and RANKING when each day has large capacity.** When running a vertex-iterative algorithm, the first jobs will be able to choose from all days. Eventually one day reaches its capacity and can no longer be chosen. Then a second day reaches its capacity, and so on. Let T_k be the (random) step at which the k^{th} day reaches its capacity. In this section, we define a fluid model which approximates the random times T_k with deterministic (fractional) times t_k. This approximation becomes exact in the limit as the number of jobs and capacity of each day grow, holding the number of days fixed. We assume familiarity with fluid limits in other contexts, and use an informal language to describe the model. Formally, the fluid limit estimate of the number of unmatched agents under procedures RANKING and RANDOM-VERTEX are $U^R(|\mathcal{J}|)$ and $U^V(|\mathcal{J}|)$, where U^R is defined by (38) and (39) and U^V is defined by (40). We plot these expressions in black in Figure 3a.

We now give the fluid limit for RANKING, with the order over days $\succ^{\mathcal{D}}$ fixed. For fixed thresholds, $t_1 \leq t_2 \leq \cdot \leq t_{|\mathcal{D}|}$, imagine running a vertex-iterative algorithm in which the job processed at step t can choose the k^{th}-ranked day if and only if $t \leq t_k$ (regardless of the choices of other jobs). Our fluid limit calculates (fractional) thresholds such that the expected number of jobs assigned to each day is equal to C.

If all days are available, a job with N feasible days selects the k^{th} ranked day with probability $f(k, N)$ given in (20). For $N \in \{1, \ldots, |\mathcal{D}|\}$, let $g(N)$ denote the fraction of jobs with N feasible days. It follows that if we select a job at random, the probability that it chooses the k^{th}-ranked day is

$$f_k = \sum_{N=1}^{|\mathcal{D}|} g(N) f(k, N).$$

Thus, if t_1 jobs are allowed to choose any day, the expected number that select the k^{th}-ranked day is $f_k t_1$. Set t_1 so that the expected number of jobs choosing the first day is equal to C:

$$f_1 t_1 = C.$$

Once the highest-ranked day has reached its capacity, each other day becomes more likely to be selected. More specifically, equations (21) and (22) from the Markov chain description of RANKING imply that each other day is effectively "promoted" one position: the second-ranked day is selected with probability f_1, the third with probability f_2, and so on. Therefore, if $t_2 - t_1$ jobs are allowed to choose any day but the first, the expected number that choose day k is $(t_2 - t_1) f_{k-1}$. Set t_2 so that the expected number of jobs choosing the second day is equal to C:

$$f_2 t_1 + f_1 (t_2 - t_1) = C.$$

Repeating this logic and setting each t_k such that the expected number of matches equal to C for each day yields the following linear system, which can be solved for $t_1, \ldots, t_{|\mathcal{D}|}$:

$$\sum_{j=1}^{k} f_j \times (t_{k-j+1} - t_{k-j}) = C, \qquad \forall k \in \{1, \ldots, |\mathcal{D}|\}. \tag{38}$$

For convenience, we define $t_0 = 0$ and $t_{|\mathcal{D}|+1} = \infty$.

Given $t_1, \ldots, t_{|\mathcal{D}|}$, we define a fluid-limit estimate of the probability of going unassigned for a random job processed in step t. Because the chance of going unassigned when k days are unavailable is equal to the chance of matching to one of the k lowest-ranked days when all days are available, we define

$$u(t) = \sum_{j=1}^{k} f_{|\mathcal{D}|+1-j} \qquad \text{for } t \in (t_k, t_{k+1}).$$

From this, we estimate the expected number of unassigned jobs after step t to be

$$U^R(t) = \int_0^t u(s)ds = \sum_{k=1}^{|\mathcal{D}|} f_{|\mathcal{D}|+1-k} \max(t - t_k, 0). \tag{39}$$

Note that the equality follows by exchanging orders of summation.

By contrast, under RANDOM-VERTEX, so long as all days are available, each job chooses a uniformly random day. Thus, in our fluid limit approximation, all days fill at the same time $t = C|\mathcal{D}|$, and the expected number of unassigned jobs is simply

$$U^V(t) = \max(t - C|\mathcal{D}|, 0). \tag{40}$$

References

Aggarwal G, Goel G, Karande C, Mehta A (2011) Online vertex-weighted bipartite matching and single-bid budgeted allocations. *Symposium on Discrete Algorithms (SODA)* 1253–1264.

Aronson J, Dyer M, Frieze A, Suen S (1995) Randomized greedy matching ii. *Random Structures & Algorithms* 6(1):55–73.

Besser B, Poloczek M (2017) Greedy matching: Guarantees and limitations. *Algorithmica* 77:201–234.

Borodin A, Nielsen M, Rackoff C (2003) (incremental) priority algorithms. *Algorithmica* 37:295–326.

Chan THH, Chen F, Wu X, Zhao Z (2018) Ranking on arbitrary graphs: Rematch via continuous lp with monotone and boundary condition constraints. *SIAM Journal on Computing* 47(4):1529–1546.

Dietzfelbinger M, Goerdt A, Mitzenmacher M, Montanari A, Pagh R, Rink M (2010) Tight thresholds for cuckoo hashing via xorsat. Abramsky S, Gavoille C, Kirchner C, Meyer auf der Heide F, Spirakis P, eds., *Automata, Languages and Programming*, volume 6198 of *Lecture Notes in Computer Science*, 213–225 (Springer Berlin Heidelberg), ISBN 978-3-642-14164-5, URL `http://dx.doi.org/10.1007/978-3-642-14165-2_19`.

Dyer M, Frieze A (1991) Randomized greedy matching. *Random Structures & Algorithms* 2(1):29–45.

Dyer M, Frieze A, Pittel B (1993) The average performance of the greedy matching algorithm. *The Annals of Applied Probability* 3(2):pp. 526–552, ISSN 10505164, URL `http://www.jstor.org/stable/2959644`.

Feldman J, Mehta A, Mirrokni V, Muthukrishnan S (2009) Online stochastic matching: Beating 1-1/e. *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science* 117–126.

Fountoulakis N, Panagiotou K (2012) Sharp load thresholds for cuckoo hashing. *Random Structures & Algorithms* 41(3):306–333, ISSN 1098-2418, URL `http://dx.doi.org/10.1002/rsa.20426`.

Frieze A, Melsted P, Mitzenmacher M (2018) An analysis of random-walk cuckoo hashing.

Frieze A, Radcliffe A, Suen S (1995) Analysis of a simple greedy matching algorithm on random cubic graphs. *Combinatorics, Probability and Computing* 4:47–66.

Goel G, Mehta A (2008) Online budgeted matching in random input models with applications to adwords. *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms* 982–991.

Goel G, Tripathi P (2012) Matching with our eyes closed. *Symposium on Fondations of Computer Science (FOCS)* 718–727.

Karande C, Mehta A, Tripathi P (2011) Online bipartite matching with unknown distributions. *ACM Symposium on Theory of Computing (STOC)*.

Karp RM, Vazirani UV, Vazirani VV (1990) An optimal algorithm for on-line bipartite matching. *Proceedings of the twenty-second annual ACM symposium on theory of computing* 352–358.

Mahdian M, Yan Q (2011) Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. *ACM Symposium on Theory of Computing (STOC)*.

Mastin A, Jaillet P (2018) Greedy online bipartite matching on random graphs. *arXiv preprint arXiv:1307.2536* .

Mehta A (2013) Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science* 8(4):265–368.

Mehta A, Saberi A, Vazirani U, Vazirani V (2007) Adwords and generalized online matching. *J. ACM* 54(5).

Poloczec M, Szegedy M (2012) Randomized greedy algorithms for the maximum matching problem with new analysis. *Symposium on Fondations of Computer Science (FOCS)* .

Tang ZG, Wu X, Zhang Y (2020) Towards a better understanding of randomized greedy matching. *Symposium on Theory of Computing (STOC)* 1097–1110.

Tinhofer G (1984) A probabilistic analysis of some greedy cardinality matching algorithms. *Annals of Operations Research* 1(3):239–254.

Wormald NC (1995) Differential equations for random processes and random graphs. *The Annals of Applied Probability* 5(4):pp. 1217–1235, ISSN 10505164, URL `http://www.jstor.org/stable/2245111`.

Wormald NC (1999) The differential equation method for random graph processes and greedy algorithms. *Lectures on approximation and randomized algorithms*, 73–155.